Completeness in abstract interpretation
Policy iteration

Damien Massé

LabSTICC, university of Brest

February 1, 2013

Fixpoint approximation

- Concrete domain: D (a poset or a lattice).
- Concrete transfer function: $\phi : D \xrightarrow{m} D$.
- Concrete semantics: $C = \operatorname{lfp} \phi$ (or $\operatorname{gfp} \phi$).

- Abstract domain: $\mathcal{D}^\sharp$ with $\gamma : D^\sharp \to D$.
- Abstract transfer function: $\phi^\sharp : D^\sharp \xrightarrow{m} D^\sharp$.
- Abstract semantics: $C^\sharp \sqsupseteq^\sharp \operatorname{lfp} \phi^\sharp$ (or $\operatorname{gfp} \phi^\sharp$).

Loss of precision

Soundness

The abstract semantics is *sound* iff $\gamma(C^\sharp) \sqsupseteq C$.

Soundness is often a consequence of:

$$\gamma \circ \phi^\sharp \sqsupseteq \phi \circ \gamma$$

Of course we cannot $\gamma(C^\sharp) = C$. The loss of precision stems from:

1. the abstraction (the best result would be $C^\sharp = \alpha(C)$);
2. the abstract transfer function (which may be **incomplete**);
3. the abstract fixpoint approximation (widening or narrowing operator).

Outline

1. Completeness of abstractions
   1. Closure operators and completeness
   2. Construction of complete domains
   3. Application to model-checking
2. Policy iteration
   1. General idea
   2. min-policies
   3. max-policies

Abstraction and closure operators

Traditionnal approach of abstract interpretation: Galois connection:

$$D \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} D^\sharp$$

with:

$$\alpha(X) \sqsubseteq Y \iff X \sqsubseteq \gamma(Y)$$

Alternative approach (to study abstractions "as abstractions"): **(upper) closure operators**:

$$\begin{aligned} \rho : D & \rightarrow D \\ X & \mapsto \gamma \circ \alpha(X) \end{aligned}$$

Closure operators

Definition

Upper closure operators are:

- monotonic: $\forall (X, Y) \in D^2, X \sqsubseteq Y \;\Rightarrow\; \rho(X) \sqsubseteq \rho(Y)$
- extensive: $\forall X \in D, X \sqsubseteq \rho(X)$
- idempotent: $\rho \circ \rho = \rho$.

Lower closure operators are monotonic, reductive and idempotent.

Properties

Closure operators can be used to abstractions without abstract domains. Let $\mathrm{uco}(D)$ (resp. $\mathrm{lco}(D)$) be the set of upper (resp. lower) closure operators on D.

Proposition

$\mathrm{uco}(D)$ is a partially ordered set: $\rho \sqsubseteq \rho'$ means that ρ' is a coarser abstraction than ρ.

Notice that $\rho \sqsubseteq \rho' \;\Rightarrow\; \rho'(D) \supseteq \rho(D)$.

Theorem

If D is a complete lattice, then so is $\mathrm{uco}(D)$.

- $(\bigsqcup \rho)(X) = \mathrm{lfp}\,\lambda Y.(X \sqcup (\bigsqcup(\rho(Y))))$
- $(\bigsqcap \rho)(X) = \bigsqcap(\rho(X))$
- *$\lambda X.X$ is the infimum, $\lambda X.\top$ is the supremum.*

$\rho \sqcap \rho'$ characterizes the *reduced product* of ρ and ρ'.

Moore families

Definition

If D is a complete lattice, a lower (resp. upper) Moore family of D is a subset L of D such that:

$$L = \mathcal{M}^l(L) = \{\sqcap X \mid X \in L\}$$

(resp. $L = \mathcal{M}^u(L) = \{\sqcup X \mid X \in L\}$).

Moore families are closed under $\sqcap$ (resp. under $\sqcup$).

Moore families and closure operators

For complete lattices, Moore families and closure operators are equivalent.

Theorem

If D is a complete lattice, then $\mathrm{uco}(D)$ and lower Moore families of D are isomorph:

1. $\forall \rho \in \mathrm{uco}(D), \rho(D)$ *is a lower Moore family.*
2. *for all Moore family L,* $\rho = \lambda X. \bigsqcap_{Y \in L, X \sqsubseteq Y} Y$ *is in* $\mathrm{uco}(D)$ *and* $\rho(D) = L$.
3. $\rho \sqsubseteq \rho' \iff \rho(D) \supseteq \rho'(D)$
4. $\bigsqcup \rho = \rho' \iff \rho'(D) = \bigcap \rho(D)$
5. $\bigsqcap \rho = \rho' \iff \rho'(D) = \mathcal{M}^l(\bigcup \rho(D))$.

Completeness

Soundness

Proposition For all $\phi : D \xrightarrow{m} D$ and $\rho \in \mathrm{uco}(D)$, we have:

$$\rho \circ \phi(X) \sqsubseteq \rho \circ \phi \circ \rho(X)$$
$$\rho(\mathrm{lfp}\phi) \sqsubseteq \mathrm{lfp}(\rho \circ \phi)$$
$$\rho(\mathrm{gfp}\phi) \sqsubseteq \mathrm{gfp}(\rho \circ \phi)$$

Completeness

Definition

1. $\rho \in \mathrm{uco}(D)$ is said to be *complete* for a monotone operator ϕ if $\rho \circ \phi = \rho \circ \phi \circ \rho$.
2. when $\rho(\mathrm{lfp}\phi) = \mathrm{lfp}(\rho \circ \phi)$, ρ is said to be lfp-complete (with respect to ϕ).
3. when $\rho(\mathrm{gfp}\phi) = \mathrm{gfp}(\rho \circ \phi)$, ρ is said to be gfp-complete (with respect to ϕ).

Notes on completeness

1. Completeness can be defined for *n*-ary operators:

$$\rho \circ \phi(x_1, \dots, x_n) = \rho \circ \phi(\rho(x_1), \dots \rho(x_n))$$

2. Completeness can be defined for a family of operators. If ρ is complete with respect to several operators, it is complete with respect to any combination of these.
3. Completeness is also called « backward completeness ». Then « forward completeness » is defined as:

$$\phi \circ \rho = \rho \circ \phi \circ \rho$$

4. With Galois connections:
   - backward completeness means: $\alpha \circ f = f^\sharp \circ \alpha$.
   - forward completeness means: $f \circ \gamma = \gamma \circ f^\sharp$.

   with $f^\sharp$ being the best abstract function ($f^\sharp = \alpha \circ f \circ \gamma$).
5. Completeness can be defined with operations over two concrete domains C and D: with $\phi : C \xrightarrow{m} D$ and $\rho \in \mathrm{uco}(C)$ and $\eta \in \mathrm{uco}(D)$, the pair $\langle \rho, \eta \rangle$ is complete for ϕ if $\eta \circ \phi = \eta \circ \phi \circ \rho$.

Examples (1)

The supremum ($\lambda x.\top$) and infimum ($\lambda x.x$) of $\mathrm{uco}(D)$ are complete for all ϕ.

All closure operators are complete with respect to $\lambda x.x$ and $\lambda x.c$ (with $c \in D$).

If $D = \wp(\mathbb{Z})$, the lattice of signs ($\{\emptyset, \{0\}, \mathbb{Z}^+, \mathbb{Z}^-, \mathbb{Z}\}$) is complete for $\lambda xy.x \times y$, but not for $\lambda xy.x + y$.

Completeness and fixpoint-completeness

Proposition

Completeness implies lfp-completeness. Completeness does not imply gfp-completeness, and fixpoint-completeness does not imply completeness.

1. lfp-complete but not complete: $D = \mathbb{N} \cup \{\omega\}$, $\phi(x) = 1 + x$, $\rho = \mathbb{N}^* \cup \{\omega\}$.
2. complete but not gfp-complete:

$$
\begin{aligned}
&D = \{[n, +\infty[\mid n \in \mathbb{N}\} \cup \{\emptyset\} \\
&\phi([n, +\infty[) = [n + 1, +\infty[\\
&\phi(\emptyset) = \emptyset \\
&\rho = \{[0, +\infty[, \emptyset\}
\end{aligned}
$$

Completeness and gfp-completeness

Proposition

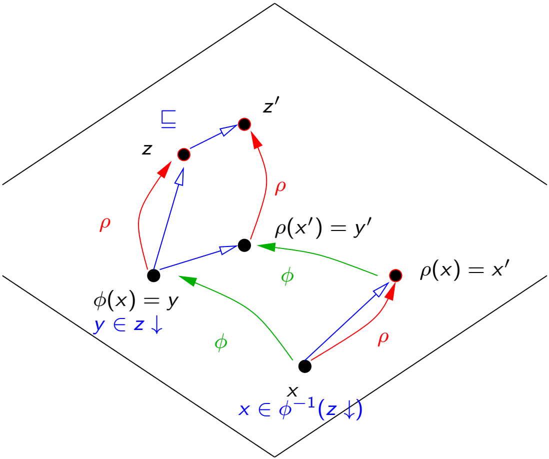
If ρ is complete w.r.t. ϕ and ρ is co-continuous, then ρ is gfp-complete.

Notes:

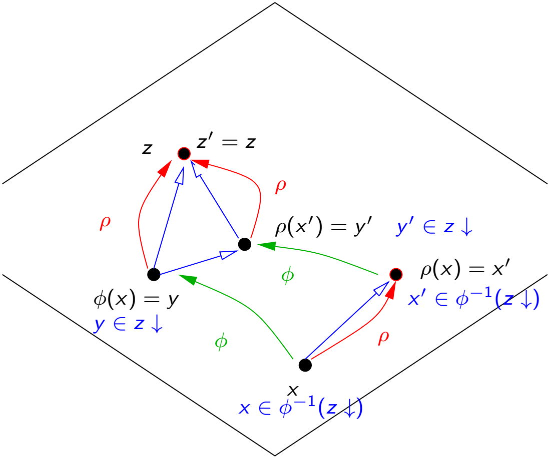
1. ρ is co-continuous means that for all decreasing chain X_i, $\rho(\sqcap X_i) = \sqcap \rho(X_i)$.
2. for lower closure operators, completeness implies gfp-completeness, completeness and continuity implies lfp-completeness.

Making abstractions complete

Let $\rho \in \mathrm{uco}(D)$ and $\phi : D \xrightarrow{m} D$.
With $x \in D$, what means $\phi \circ \rho \circ \phi(x) = \phi \circ \rho(x)$?



(here $X \uparrow = \{Y \mid Y \sqsupseteq X\}$ and $X \downarrow = \{Y \mid Y \sqsubseteq X\}$).



$\exists x' \in \rho$ such that $x' \sqsupseteq x$ and $x' \in \phi^{-1}(\rho(\phi(x))\downarrow)$ is sufficient.

Equivalence of completeness

Lemma

ρ is complete with respect to ϕ iff:

$$\begin{aligned}&\forall z \in \rho,\\&\quad \forall x \in \phi^{-1}(z{\downarrow}),\\&\qquad \exists x' \in \rho \text{ s.t. } x \sqsubseteq x' \text{ and } x' \in \phi^{-1}(z{\downarrow})\end{aligned}$$

So, for all z in ρ, "maximal" elements of $\phi^{-1}(z{\downarrow})$ must be un ρ.

Construction of complete operators

Starting from an operator ρ, if ρ is not complete wrt. ϕ:

$$
\begin{array}{l}
\exists z \in \rho, \\
\quad \exists x \in \phi^{-1}(z\downarrow), \\
\quad\quad \forall x', (x \sqsubseteq x' \text{ and } x' \in \phi^{-1}(z\downarrow)) \Rightarrow x' \notin \rho
\end{array}
$$

We can make ρ complete:

1. by removing z;
2. or by adding an x' in ρ.

Example

With the sign abstraction ρ and $\phi(X) = \{x+1 \mid x \in X\}$.

- with $z = \emptyset$, $\phi^{-1}(z\downarrow) = \{\emptyset\}$, ok.
- with $z = \{0\}$, $\phi^{-1}(z) = \{\emptyset, \{-1\}\}$ not ok $\rightarrow$ remove $\{0\}$ or add $\{-1\}$.
- with $z = \mathbb{Z}^-$, $\phi^{-1}(z\downarrow) = \wp(\mathbb{Z}^{-*})$, ok.
- with $z = \mathbb{Z}^{-*}$, not ok, remove $\mathbb{Z}^{-*}$ or add $]-\infty, -2]$
- with $z = \mathbb{Z}^+$, not ok, remove it or add $[-1, +\infty[$.
- with $z = \mathbb{Z}^{+*}$, ok.
- with $z = \mathbb{Z}^*$, not ok, remove it or add $]-\infty, -2] \cup [0, +\infty[$.
- with $z = \mathbb{Z}$, ok.

Easier case: ϕ continuous

When ϕ is continuous, $\phi^{-1}(z{\downarrow})$ is bounded by its maximal elements.

Lemma

Let $\phi : D \xrightarrow{c} D$ and $z \in D$. If $x \in \phi^{-1}(z{\downarrow})$ then there exists $y \in \max(\phi^{-1}(z{\downarrow}))$ such that $x \sqsubseteq y$.

Thus ρ is complete w.r.t. ϕ iff:

$$\forall z \in \rho,\ \max(\phi^{-1}(z{\downarrow})) \subseteq \rho$$

Note: with $\phi : C \xrightarrow{c} D$, the pair $\langle \rho, \eta \rangle$ with $\rho \in \mathrm{uco}(C)$ and $\eta \in \mathrm{uco}(D)$ is complete w.r.t. ϕ ($\eta \circ \phi \circ \rho = \eta \circ \phi$ iff:

$$\forall z \in \eta,\ \max(\phi^{-1}(z{\downarrow})) \subseteq \rho$$

Removing elements

We define:

$$L_\phi(\rho) = \{z \in D \mid \max(\phi^{-1}(z\downarrow)) \subseteq \rho\}$$

Lemma

$L_\phi(\rho)$ is a Moore family.

Sketch of proof : let $Z \subseteq L_\phi(\rho)$ (with $Z \neq \emptyset$), and $w = \sqcap Z$.
Let $x \in \max(\phi^{-1}(w\downarrow))$. Then for all $z \in Z$, $x \in \phi^{-1}(z\downarrow)$, so $x \sqsubseteq m_z$ with $m_z \in \max(\phi^{-1}(z\downarrow))$. Since $\phi(\sqcap_{z\in Z} m_z) \sqsubseteq w$, we have $\sqcap_{z\in Z} m_z \in \phi^{-1}(w\downarrow)$, and by maximality, $\sqcap_{z\in Z} m_z = x$. Thus $x \in \rho$, which proves that $w \in L_\phi(\rho)$.
(when $Z = \emptyset$, $x = \top$, hence $x \in \rho$).

Adding elements

We define:

$$R_\phi(\rho) = \mathcal{M}^l(\bigcup_{z \in \rho} \max(\phi^{-1}(z\downarrow)))$$

Theorem

1. $L_\phi(\rho) \circ \phi \circ \rho = L_\phi(\rho) \circ \phi$ *(i.e.* $\langle \rho, L_\phi(\rho) \rangle$ *is complete w.r.t.* ϕ*);*
2. $\rho \circ \phi \circ R_\phi(\rho) = \rho \circ \phi$ *((i.e.* $\langle R_\phi(\rho), \rho \rangle$ *is complete w.r.t.* ϕ*).*

Sketch of proof:

1. $\forall x$, if $z = (L_\phi(\rho) \circ \phi)(x)$, then $x \in \phi^{-1}(z\downarrow)$, so $x \sqsubseteq y$ st. $y \in \max(\phi^{-1}(z\downarrow)) \subseteq \rho$, so $\rho(x) \sqsubseteq y$ so $(L_\phi(\rho) \circ \phi \circ \rho)(x) \sqsubseteq z$.
2. similar but $y \in R_\phi(\rho)$.

Corollary

Corollary: for all $(\rho, \eta) \in \mathrm{uco}(D)$, the three propositions are equivalent:

1. $\eta \circ \phi \circ \rho = \eta \circ \phi$
2. $L_\phi(\rho) \sqsubseteq \eta$
3. $\rho \sqsubseteq R_\phi(\eta)$

Therefore:

1. we have a Galois connection: $\mathrm{uco}(D) \underset{L_\phi}{\overset{R_\phi}{\leftrightarrows}} \mathrm{uco}(D)$
2. L_ϕ is additive, and R_ϕ is coadditive.

Absolute complete core

Definition: the *absolute complete core* of ρ for ϕ, when it exists, is the minimal closure operator $\mathcal{C}_\phi(\rho)$ greater than ρ and complete wrt ϕ.

Theorem: if ϕ is continuous, then for any $\rho \in \mathrm{uco}(D)$, the absolute complete core of ρ for ϕ exists and is defined as:

$$\mathcal{C}_\phi(\rho) = \mathrm{lfp}\mathcal{L}_\phi^\rho$$

with

$$\mathcal{L}_\phi^\rho = \lambda\eta.\rho \sqcup L_\phi(\eta)$$

Furthermore, $\mathcal{L}_\phi^\rho$ is continuous (since L_ϕ is additive) so the fixpoint is reached after (at most) ω iterations.

Absolute complete shell

Definition: the *absolute complete shell* of ρ for ϕ, when it exists, is the maximal closure operator $\mathcal{S}_\phi(\rho)$ less than ρ and complete wrt ϕ.

Theorem: if ϕ is continuous, then for any $\rho \in \mathrm{uco}(D)$, the absolute complete shell of ρ for ϕ exists and is defined as:

$$\mathcal{S}_\phi(\rho) = \mathrm{gfp}\mathcal{R}_\phi^\rho$$

with

$$\mathcal{R}_\phi^\rho = \lambda\eta.\rho \sqcup R_\phi(\eta)$$

Furthermore, $\mathcal{R}_\phi^\rho$ is cocontinuous (since R_ϕ is additive) so the fixpoint is reached after (at most) ω iterations.

Example

With $D = \wp(\mathbb{Z})$, let $\rho = \{\emptyset, \mathbb{Z}\} \cup \{]-\infty, n] \mid n \in \mathbb{Z}\}$.
With $\phi = \lambda X.\{x^2 \mid x \in X\}$, we have:

$$\max \phi^{-1}(]-\infty, n] \downarrow) = \begin{cases} \emptyset & \text{if } n < 0 \\ [-\lfloor\sqrt{n}\rfloor, \lfloor\sqrt{n}\rfloor] & \text{if } n \geq 0 \end{cases}$$

Hence,

$$\mathcal{C}_\phi(\rho) = \{\emptyset, \mathbb{Z}\} \cup \{]-\infty, n] \mid n < 0\}$$
$$\mathcal{R}_\phi(\rho) = \{\emptyset\} \cup \{[-m, n] \mid |n| \leq m \leq +\infty\}$$

Application to model-checking

Transition system: (Σ, τ), with $\tau \in \Sigma \times \Sigma$.

Definition: classical *predicate transformers* from $\wp(Sigma)$ to $\wp(\Sigma)$:

$$
\begin{aligned}
\mathrm{pre}[\tau](Y) &= \{\sigma \in \Sigma \mid \exists \sigma' \in Y, (\sigma, \sigma') \in \tau\} \\
\widetilde{\mathrm{pre}}[\tau](Y) &= \{\sigma \in \Sigma \mid \forall \sigma' \in \Sigma, (\sigma, \sigma') \in \tau \Rightarrow \sigma' \in Y\} \\
\mathrm{post}[\tau](X) &= \{\sigma' \in \Sigma \mid \exists \sigma \in X, (\sigma, \sigma') \in \tau\} \\
\widetilde{\mathrm{post}}[\tau](X) &= \{\sigma' \in \Sigma \mid \forall \sigma \in \Sigma, (\sigma, \sigma') \in \tau \Rightarrow \sigma \in X\}
\end{aligned}
$$

We may omit $[\tau]$.

Predicate transformers : basic results

Lemma: $\forall (X, Y) \in \wp(\Sigma)^2$:

1. $\text{post}(X) \subseteq Y \iff X \subseteq \widetilde{\text{pre}}(Y)$
2. $\text{pre}(Y) \subseteq X \iff Y \subseteq \widetilde{\text{post}}(Y)$

Proposition: given three sets of states I, F and S:

1. the set of states reachable from I (forward collecting semantics) is $\text{lfp}\lambda X.(I \cup \text{post}(X))$.
2. the set of states which *may* (backward collecting semantics) reach F is $\text{lfp}\lambda X.(I \cup \text{pre}(X))$.
3. the set of states which *will* reach F is $\text{lfp}\lambda X.(I \cup \widetilde{\text{pre}}(X))$.
4. the set of states which *may* « stay » in S is $\text{gfp}\lambda X.(S \cap \text{pre}(X))$.
5. the set of states which *will* « stay » in S is $\text{gfp}\lambda X.(S \cap \widetilde{\text{pre}}(X))$.

Partitions of states

Standard model-checking relies on abstract structures on partitions of states:

Let A be a partition of Σ. The abstraction is $\wp(\Sigma) \xleftarrow[\alpha]{\gamma} \wp(A)$ with

$$\alpha(X) = \{S \in A \mid S \cap X \neq \emptyset\}$$
$$\gamma(X) = \cup X$$

The upper closure operator $\rho = \gamma \circ \alpha$ is then

$$\rho(X) = \{\cup E \mid E \in A \wedge X \cap E \neq \emptyset\}$$

On A, we can define a (abstract) transition system $(A, \tau^\sharp)$. An example of $\tau^\sharp$:

$$(S, S') \in \tau^\sharp \iff \exists \sigma \in S, \exists \sigma' \in S', (\sigma, \sigma') \in \tau$$

With this example:

$$\mathrm{pre}[\tau^\sharp] = \alpha \circ \mathrm{pre}[\tau] \circ \gamma$$
$$\mathrm{post}[\tau^\sharp] = \alpha \circ \mathrm{post}[\tau] \circ \gamma$$

Preservation

From the soundness of abstraction, we can deduce that (for example):

$$\alpha(\mathrm{lfp}\lambda X.(I \cup \mathrm{post}[\tau](X))) \subseteq \mathrm{lfp}\lambda S.(\alpha(I) \cup \mathrm{post}[\tau^\sharp](S))$$

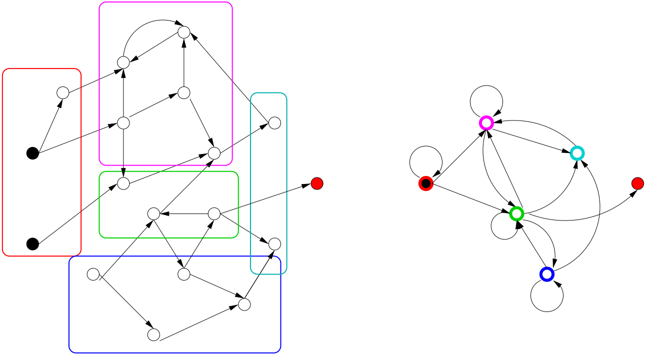
which means that each concrete state reachable from I can be associated to an abstract state reachable from $\alpha(I)$ in the abstract structure (this property is known as *weak preservation*).

A *complete* abstraction would imply:

$$\alpha(\mathrm{lfp}\lambda X.(I \cup \mathrm{post}[\tau](X))) = \mathrm{lfp}\lambda S.(\alpha(I) \cup \mathrm{post}[\tau^\sharp](S))$$

This property would be known as *strong preservation*.

Example



In this example, weak preservation is satisfied, but not strong preservation.

Refinement in model-checking

Any partition can be associated to an abstract domain. But an abstract domaine does not always induce a partition. But we can generate a new partition from a refined abstract domain.

Proposition: let A be a partition of Σ, and ρ the associated closure ($\rho(X) = \{S \in A \mid S \cap X \neq \emptyset\}$). From $\rho' \sqsubseteq \rho$, we can deduce a new partition A':

$$S \in A' \iff \exists \sigma \in \Sigma, \rho'(\{\sigma\}) = S$$

Then A' is finer than A.

Hence we can make refinement on partitions.

Completeness of the abstraction

Notice that completeness means here:

$$\alpha \circ \mathrm{post}[\tau] = \mathrm{post}[\tau^\sharp] \circ \alpha$$

which is equivalent to:

1. $\rho \circ \mathrm{post}[\tau] \circ \rho = \rho \circ \mathrm{post}[\tau]$ (backward completeness);
2. $\rho \circ \widetilde{\mathrm{pre}}[\tau] \circ \rho = \widetilde{\mathrm{pre}}[\tau] \circ \rho$ (hence the notion of *forward completeness*).

Constructing complete abstraction

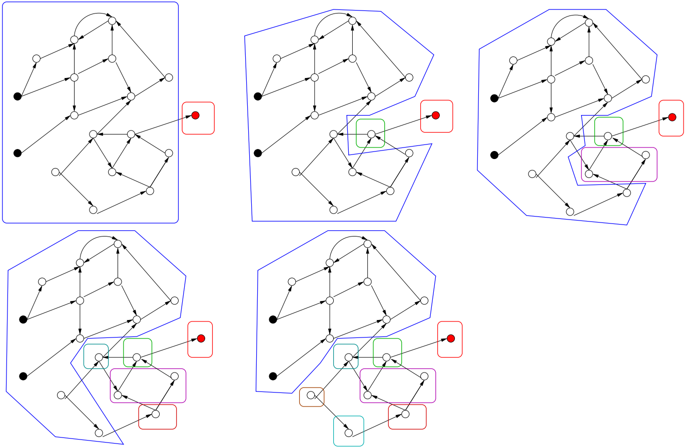
Since $\mathrm{post}[\tau]$ is continuous, the absolute complete shell of ρ exists and is:

$$\mathcal{S}_{\mathrm{post}[\tau]}(\rho) = \mathrm{gfp}\lambda\eta.(\rho \sqcup \mathcal{M}^{I}(\bigcup_{X\in\rho} \mathsf{max}(\mathrm{post}[\tau]^{-1}(X\downarrow))))$$

We can see that: $\mathsf{max}(\mathrm{post}[\tau]^{-1}(X\downarrow)) = \widetilde{\mathrm{pre}}(X)$.

Successive refinements

The successive iterations give successive refinements of the initial partition.



This approach gives a theoretical basis of CEGAR (Counterexample guided abstraction refinement) where the refinements are limited to counterexample traces.

Outline

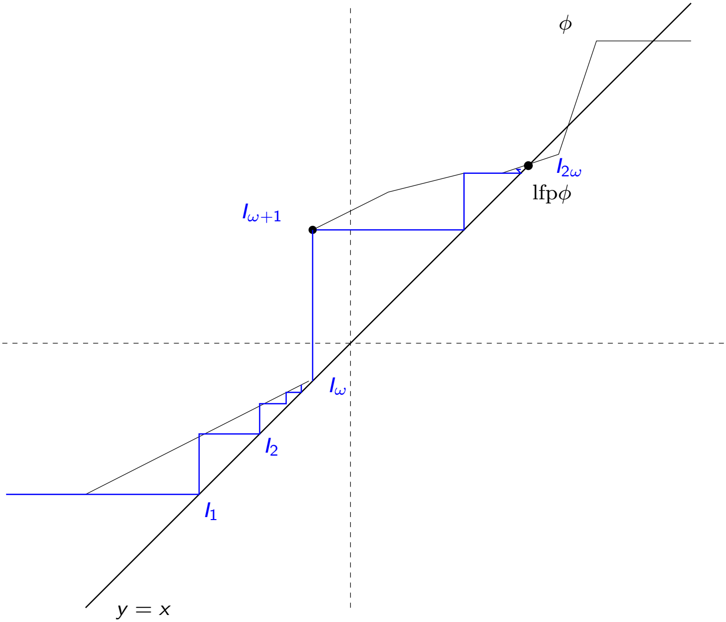
1. Completeness of abstractions
   1. Closure operators and completeness
   2. Construction of complete domains
   3. Application to model-checking
2. Policy iteration
   1. General idea
   2. min-policies
   3. max-policies

Fixpoint approximation by widenings/narrowings

Common approach (cf Cousot's thesis) to approximate fixpoints on infinite-height lattices.
However this approach loses precision:

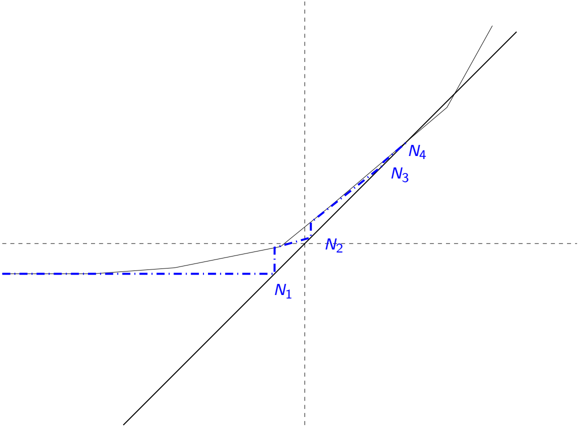
- widenings (for lfp) are non-monotonic, imprecise;
- narrowings are worse.

More generally, Kleene iterations are a slow and inefficient way to solve an equation, when there exists direct (algebraic) methods, or faster methods.



ϕ
$I_{2\omega}$
lfpϕ
$I_{\omega+1}$
I_ω
I_2
I_1
$y = x$

Newton method

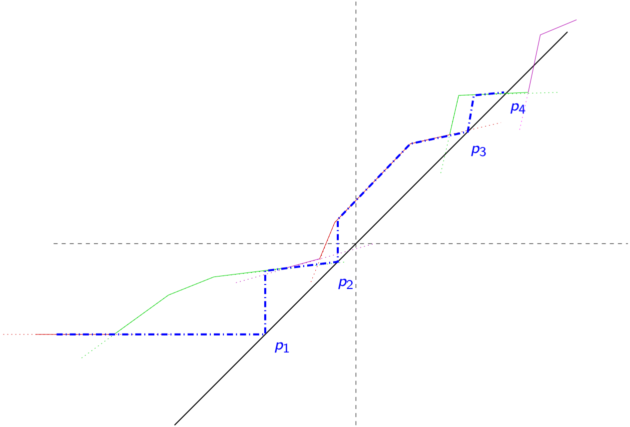


However, Newton method does not guarantee to get the least fixpoint (even starting from $-\infty$). We need:

- a *convex* function ($f = \max\{\text{tangents}(f)\}$);
- and a finite number of iterations (e.g. piecewise linear function).

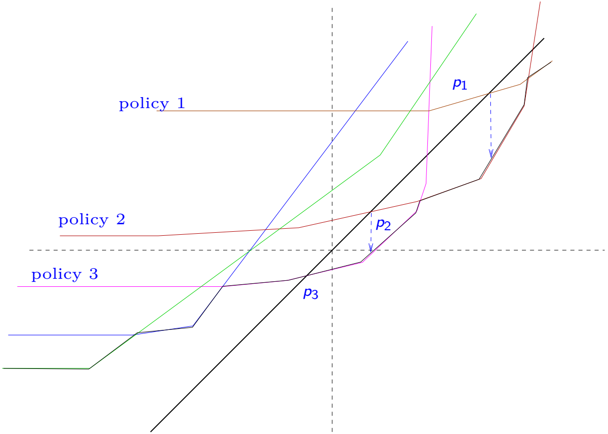
With non-convex function

We could consider $f = \max f_i$ where each f_i is *concave*. If we can compute the next fixpoint of f_i at each iteration, we can obtain the fixpoint of f.



Alternative computation (from "above")

Here $f = \min f_i$ where each f_i is *convex*. But we may not approximate the least fixpoint.



Policy iteration

Policy iteration (or strategy iteration) comes in two flavours:

1. From $\forall x, \phi(x) = \min \phi_i(x)$, we have:

$$\mathrm{lfp}\, \phi = \min \mathrm{lfp}\, \phi_i$$

   - ϕ_i are the min-policies;
   - soundness is trivial;
   - policy initialisation and improvement modify the precision

2. From $\forall x, \phi(x) = \max \phi_i(x)$, we have:

$$\mathrm{lfp}\, \phi = \mathrm{lfp}\, \lambda x.(\mathrm{lfp}_{\sqsupseteq x} \phi_{i(x)})$$

   (where $i(x)$ is such that $\phi_{i(x)}(x) = \phi(x)$).

   - ϕ_i are the max-policies (strategies);
   - soundness is tricky, and related to policy improvement;
   - precision is automatic.

Context

Policy iterations can be used to compute the exact abstract fixpoint. For obvious reasons, they cannot be applied for any domain and abstract functions:

- specific numerical domains (e.g. weakly relational domains) appear to be good choices :
  - notion of *convexity*;
  - finite number of equations.
- programs must be adapted to the abstract domain (e.g. affine programs).

Affine programs

An affine program is defined by $(N, E, \mathbf{st})$ where

- N is the finite set of program points;
- $E \subseteq N \times \mathbf{Stmt} \times N$ transitions labelled by *statements*;
- $\mathbf{st}$ initial program point.

Statements are transitions which can include:

- affine guards $A\mathbf{x} + b \geq 0$ on the program variables $\mathbf{x}$
- affine assignments $\mathbf{x} := A\mathbf{x} + b$.

More generally, we can define a statement $(Q, \mathbf{q})$ as linear constraints between the variables before $(\mathbf{x})$ and after $(\mathbf{x}')$ the transition:

$$(Q) . \begin{pmatrix} \mathbf{x} \\ \mathbf{x}' \end{pmatrix} \leq (\mathbf{q})$$

Template polyhedral domain

Most common example of **weakly relational domain**.
Abstraction of $\wp(\mathbb{R}^n)$ relative to a template constraint matrix $T \in \mathbb{R}^{m\times n}$:

$$\wp(\mathbb{R}^n) \underset{\alpha_T}{\overset{\gamma_T}{\leftrightarrows}} (\mathbb{R} \cup \{-\infty, +\infty\})^m$$

with $\gamma_T(\rho) = \{x \in \mathbb{R}^n \mid Tx \leq \rho\}$.

Example: octagons with two variables: $T = \begin{pmatrix} 0 & 1 \\ 0 & -1 \\ 1 & 0 \\ -1 & 0 \\ 1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & -1 \end{pmatrix}$

$\rightarrow$ 8 “abstract” variables (C_y, C_{-y}, $\ldots$).

Abstraction and semantic equation

The abstraction function $\alpha_T : \wp(\mathbb{R}^n) \rightarrow \mathbb{R} \cup \{-\infty, +\infty\})^m$ is defined as:

$$[\alpha_T(X)]_i = \max\{T_i x \mid x \in X\}$$

If X is a convex polyhedron, this function can be computed using linear programming.

Proposition (abstract transition)

Given a set of states (at a program point $n \in N$) represented by a polyhedron $P : TX \leq \rho$, the abstraction of a set of successor states after one affine transition $(Q, \mathbf{q})$ from n to n' is represented by the polyhedron $P' : TX \leq \rho'$ where:

$$[\rho']_i = \max\{T_i x' \mid \begin{pmatrix} & Q \\ T & 0 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{x}' \end{pmatrix} \leq \begin{pmatrix} \mathbf{q} \\ \rho \end{pmatrix}\}$$

Notice that any modification of ρ only changes the right-hand side of the linear program.

Duality result

- $[\rho']_i = -\infty$ if $\begin{pmatrix} & Q \\ T & 0 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{x}' \end{pmatrix} \leq \begin{pmatrix} \mathbf{q} \\ \rho \end{pmatrix}$ is unsatisfiable.
- otherwise:

$$[\rho']_i = \min\{\left(\mathbf{q}^\mathsf{T} \rho^\mathsf{T}\right) \boldsymbol{\lambda} \mid \boldsymbol{\lambda} \geq 0 \wedge \begin{pmatrix} Q^\mathsf{T} & T^\mathsf{T} \\ & 0 \end{pmatrix} (\boldsymbol{\lambda}) = \begin{pmatrix} 0 \\ T_i^\mathsf{T} \end{pmatrix}\}$$

Here, any modification ρ only changes the objective function of the linear program, and not the polytope.

Example

One transition, with the guard: $x + y \leq 10$ and the assignments $x' = -2y$, $y' = x - y + 3$. For the octagon domain, the abstraction of the **pre** operator on the transition gives:

$$C_x = \min(\psi, \phi)$$

with

- $\psi = -\infty$ iff the set of constraints $\{\ x + y - 10 \leq 0,\ x - y + 3 \leq C_y,\ -x + y - 3 \leq C_{-y},\ -2y \leq C_x,\ 2y \leq C_{-x},\ x - 3y + 3 \leq C_{x+y},\ -x - y - 3 \leq C_{x-y},\ x + y + 3 \leq C_{-x+y},\ -x + 3y - 3 \leq C_{-x-y}\}$ is unsatisfiable.
- $$\begin{aligned}\phi = \min\{&10\lambda_0 + \lambda_1(C_y - 3) + \lambda_2(C_{-y} + 3) + \lambda_3 C_x + \lambda_4 C_{-x} + \lambda_5(C_{x+y} - 3)\\ &+\lambda_6(C_{x-y} + 3) + \lambda_7(C_{-x+y} - 3) + \lambda_8(C_{-x-y} + 3)\\ &|\boldsymbol{\lambda} \geq 0 \wedge \lambda_0 + \lambda_1 - \lambda_2 + \lambda_5 - \lambda_6 + \lambda_7 - \lambda_8 = 1\\ &\wedge \lambda_0 - \lambda_1 + \lambda_2 - 2\lambda_3 + 2\lambda_4 - 3\lambda_5 - \lambda_6 + \lambda_7 + 3\lambda_8 = 0\}\end{aligned}$$

Equations

Vertex principle of linear programming

If there is a minimum value of the linear program, it occurs at one or more vertices.

Thus, if $[\rho']_i$ is not $-\infty$, it can be defined as the minimum of a finite number of affine function on ρ (one for each vertex of the polytope).

Proposition

$[\rho']_i = \min(\psi_i(\rho), \phi_i(\rho))$ where

- ψ_i is monotonic and its image is in $\{-\infty, +\infty\}$
- ϕ_i is the minimum of a (finite) number of several (monotonic) affine functions.

Example

$$\phi = \min\{10\lambda_0 + \lambda_1(C_y - 3) + \lambda_2(C_{-y} + 3) + \lambda_3 C_x + \lambda_4 C_{-x} + \lambda_5(C_{x+y} - 3)$$
$$+\lambda_6(C_{x-y} + 3) + \lambda_7(C_{-x+y} - 3) + \lambda_8(C_{-x-y} + 3)$$
$$|\boldsymbol{\lambda} \geq 0 \wedge \lambda_0 + \lambda_1 - \lambda_2 + \lambda_5 - \lambda_6 + \lambda_7 - \lambda_8 = 1$$
$$\wedge \lambda_0 - \lambda_1 + \lambda_2 - 2\lambda_3 + 2\lambda_4 - 3\lambda_5 - \lambda_6 + \lambda_7 + 3\lambda_8 = 0\}$$

With $C_{x+y} = 10$ and $C_x = C_{-x} = \ldots = C_{-x-y} = +\infty$, the optimal solution is:

$$\lambda_5 = 0.25 \quad \lambda_0 = 0.75 \quad \lambda_i = 0 \text{ for } i \notin \{0, 5\}$$

which gives the affine expression:

$$6.75 + 0.25 C_{x+y}$$

Hence we have $\phi = \min(6.75 + 0.25 C_{x+y}, \ldots)$. The number of affine expressions is exponential, hence we will try to compute them lazily.

Result

The abstract semantics of the program is the least solution of a system of equations of the form:

$$x_i = \max(\min(\psi_i^1, \phi_i^1), \min(\psi_i^2, \phi_i^2), \ldots)$$

where ϕ_i^j are monotonic and their images are in $\{-\infty, +\infty\}$, and ψ_i^j are the minimum of a finite number of affine functions.
Notice that ϕ_i^j and affine functions are convex and concave. However, the min operator is concave, and the max operator is convex.

min-policies: policy selection

With min-policies, we construct a decreasing chain of post-fixpoints (each one being the lfp of a policy).

- Initial post-fixpoint: any post-fixpoint ρ_0, computed e.g. with Kleene iterations and widenings.
- Policy selection: from ρ_k, compute $\psi_i^j(\rho_k)$. If the result is $-\infty$, select $-\infty$, otherwise compute $\phi_i^j(\rho_k)$ and select the optimal vertex.

min-policies: fixpoint computation

Policy selection gives an equation system of the form:

$$x_i = \max(a_1^i(x), a_2^i(x), \ldots)$$

where each a_j^i is an affine and monotonic function. We can rewrite the system as constraints:

$$x_i \geq a_j^i(x) \quad \forall i, j$$

The result is a polytope, whose minimum (e.g. the point minimizing $x_1 + x_2 + \ldots + x_n$, for finite components) is the least fixpoint of the system. Hence we can compute it by solving a linear program.
The result is a new post-fixpoint ρ_{k+1}, which can be used to compute a new policy.
The process terminates (the total number of policies is finite), but may not give the lfp of the system. However, any intermediate result is sound.

max-policies: policy selection

With max-policies, we construct a increasing chain of pre-fixpoints.

- Initial pre-fixpoint: $-\infty$.
- Policy selection: from ρ_k, compute $\min(\psi_i^j(\rho_k), \phi_i^j(\rho_k))$. Select the "best" transition (which gives the maximum).

max-policies: fixpoint computation

Policy selection gives an system of equations of the form $x_i = \phi_i^j$ where ϕ_i^j is a linear program of the form:

$$\min\{\left(\mathbf{q}^\mathsf{T}\rho^\mathsf{T}\right)\boldsymbol{\lambda} \mid \boldsymbol{\lambda} \geq 0 \wedge (A)\boldsymbol{\lambda} = (\boldsymbol{b})\}$$

Theorem

If the policy improvement step is "lazy" (i.e. keeps the current policy as much as possible), and the solution is finite, then the least solution of the system greater than ρ_k is the greatest finite solution of the system:

$$x_i \leq \phi_i^j$$

Intuition: this system describes a convex set of (strict) pre-fixpoint for the semantics equations, including ρ_k. The "next" fixpoint is the greatest element of this convex set.

However, the proof is a bit complicated (see Gawlitza and Seidl, ACM TPLS 2011) and is done by induction over the successive policies.

Fixpoint computation

We can rewrite the system as constraints:

$$x_i \leq T_i \boldsymbol{y'}$$

$$(A)\begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{y'} \end{pmatrix} \leq \begin{pmatrix} \boldsymbol{q} \\ \boldsymbol{x} \end{pmatrix}$$

The result is a polytope, whose finite maximum (e.g. the point maximizing $x_1 + x_2 + \ldots + x_n$, for the components which are not $+\infty$ or $-\infty$) is the least fixpoint of the system. Hence we can compute it by solving a linear program.

The result is a new post-fixpoint ρ_{k+1}, which can be used to compute a new policy.

The process terminates (the total number of policies is finite), and gives the lfp of the abstract semantics. Any intermediate result is **not** sound.

gfp computation

Policy iteration can be used to compute overapproximations of gfp (in replacement of narrowings), but:

- min-policies become max-policies, and vice-versa.
- max-policies can only be used if we can prove that we reach the gfp. Intermediate results are **not** sound.
- min-policies computes the abstract semantics and any intermediate result is sound.

This approach can be used to prove the termination of a program (or find an over-approximation of the non-terminating states).

Exemple

With only one loop:

real x,y;
while (x+y<=10) { x=-2y // y=x-y+3; }

#	Strategy	Solution
1	$C_{x+y} = 10$	$x + y \leq 10$
2	$C_x = 6.75 + 0.25C_{x+y},\ C_{x+y} = 10,$ $C_{x-y} = 3.5 + C_{x+y}/2$	$x \leq 9.25,\ x + y \leq 10$ $x - y \leq 8.5$
3	$C_x = 6.75 + 0.25C_{x+y},\ C_{x+y} = 10,$ $C_{x-y} = 3.5 + C_{x+y}/2,\ C_{-y} = 0.5C_x,$ $C_{-x-y} = 3 + C_{x-y}$	$x \leq 9.25,\ -4.625 \leq y$ $-11.5 \leq x + y \leq 10$ $x - y \leq 8.5$
4	$C_x = 6.75 + 0.25C_{x+y},\ C_{x+y} = 10,$ $C_{x-y} = 3.5 + C_{x+y}/2,\ C_{-y} = 0.5C_x,$ $C_{-x-y} = 3 + C_{x-y}, C_y = 3.25 + 0.25C_{-x-y}$ $C_{y-x} = 3 + C_{-y},\ C_{-x} = 3 + 0.5C_{-x-y} + 0.5C_{-y}$	$-9.5625 \leq x \leq 9.25$ $-4.625 \leq y \leq 6.125$ $-11.5 \leq x + y \leq 10$ $-7.625 \leq x - y \leq 8.5$
5	$C_x = -3 + 0.5C_{-x-y} + 0.5C_y,$ $C_{x+y} = -3 + C_{-x+y},\ C_{x-y} = -3 + C_y$ $C_{-y} = 0.5C_x,\ C_{-x-y} = 3 + C_{x-y},$ $C_y = 0.5C_{-x},\ C_{y-x} = 3 + C_{-y},$ $C_{-x} = 3 + 0.5C_{-x-y} + 0.5C_{-y}$	$x = -1.5,\ y = 0.75$

The program terminates from any state $\neq (-1.5, 0.75)$.

Issues

1. Complexity of the approach: exponential in theory, and in practice?
2. Selection of the template?

Policy iteration has been extended to quadratic zone domains (an extension of polyhedral template with quadratic constraints), using semi-definite programming. Its extension to more complex domains (e.g. convex polyhedra) seems difficult.

Bibliography

- On the construction of complete operators:
  R. Giacobazzi, F. Ranzato and F. Scozarri, *Making abstraction complete*, Journal of the ACM, 47(2):361–416, 2000.
- On the application of completeness to abstract model-cheking:
  - R. Giacobazzi and E. Quintarelli, *Incompleteness, counterexamples and refinements in abstract model-checking*, Proceedings of SAS'01, LNCS 2126, 2001.
  - F. Ranzato and F. Tapparo, *Strong preservation as completeness in abstract interpretation*, Proceedings of ESOP'04, LNCS 2986, 2004.

Bibliography (2)

On policy iterations:

- A. Costan et al., *A Policy Iteration Algorithm for Computing Fixed Points in Static Analysis of Programs*, Proceedings of CAV'05, LNCS 3576, 2005.
- S. Gaubert et al., *Static Analysis by Policy Iteration on Relational Domains*, Proceedings of ESOP'07, LNCS 4421, 2007,
- A. Adje et al., *Coupling Policy Iteration with Semi-definite Relaxation to Compute Accurate Numerical Invariants in Static Analysis*, Proceedings of ESOP'10, LNCS 6012, 2010.
- T. Gawlitza and H. Seidl, *Solving systems of rational equations through strategy iteration*, ACM Trans. Program. Lang. Syst. vol 33, 2011.
- T. Gawlitza et al., *Abstract interpretation meets convex optimization*, Journal of Symbolic Computation, Vol 47 issue 12, Sept. 2012.
- D. Massé, *Proving Termination by Policy Iteration*, NSAD 2012.